

Reducing waiting and idle time for a group of jobs in the grid computing

Mahdi S. Almhanna, Firas Sabah Al-Turaihi, Tariq A. Murshedi

Department of Information Networks, College of Information Technology, University of Babylon, Babylon, Iraq

Article Info

Article history:

Received Sep 8, 2022
Revised Nov 25, 2022
Accepted Dec 28, 2022

Keywords:

Big data
Cloud computing
Data grid
Grid computing
Distributed systems
Load balancing algorithm
Weighted round-robin

ABSTRACT

Johnson's rule is a scheduling method for the sequence of jobs. Its primary goal is to find the perfect sequence of functions to reduce the amount of idle time, and it also reduces the total time required to complete all functions. It is a suitable method for scheduling the purposes of two functions in a specific time-dependent sequence for both functions and where the time factor is the only parameter used in this way. Therefore, it is not suitable for scheduling work for computers network, where there are many factors affecting the completion time such as CPU speed, memory, bandwidth, and size of data. In this research, Johnson's method will adopt by adding many factors that affect the completion time of the work so that it becomes suitable for the site's job scheduling purposes to reduce the waiting and idle time for a group of jobs.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Mahdi S. Almhanna
Department of Information Security, College of Information Technology, University of Babylon
Babylon, Iraq
Email: mahdi.almhanna@uobabylon.edu.iq

1. INTRODUCTION

Johnson's rule is a scheduling method for the sequence of jobs [1]. When the database saves on multiple sites across various locations, it's called distributed database [2], this database stores in multiple computers existing on the same sites, or it can share through a network of computers connected to each other [2], [3]. This system comprises a database distributed over unrelated sites with some [4], [5] because it does not share any of the physical system components. Administrators can have control over these databases, as they can distribute sets of similar data through the replication process, or even sets of different data, and in both cases, this can be by distributing them to multiple physical sites on network servers or decentralized computers on the internet, such as by allowing tasks to be performed on multiple devices and end-user performance can have improved by using distributed databases [6].

The problem we may have is that in some requests we may need some data on one server called A which contains part of the database and the other data keep on another server, let's call it B, the data on server A may be an input to that data on server B. Such a dilemma may require careful scheduling of these requests so that processing is in the least amount of time and can get by maximizing interference in servers work (reducing the total elapsed time as much as possible) and reducing the idle time as much as possible from both servers [7], [8]. According to Johnson's procedure [1] which is based on the processing time of requests only, has improved. It has developed by adding some other variables in order to simulate the actual and practical reality of this method.

In other words, the dilemma is how to find the best order in which you must implement a set of tasks to execute on different devices for effective use of the facilities available to reach greater throughput through reduces the total time elapsed (the time of starting processes first job to the end of processes of the last one).

For example, suppose that on two machines there are two tasks need processing. Then the function can enumerate by the counting method. However, if the number of required functions and the machines to be performed increases, the problem will be more complicated. So, the traditional method of counting will be inappropriate and useless. Suppose we have M number of machines and N number of functions. Then we have around $N!$ to the power of M sequences. For example, if $N=6$ and $M=5$, then $(6!)^5=193, 491, 763, 200,000$ sequences we have. It takes a long time to search for all sequences and choose the optimal sequence among all sequences. Next, we are looking for an easier way to find the optimal sequence. Johnson's procedure is a good way to explain the interference between machines and to reduce the idle time of those machines, but this method relies on one parameter to calculate that relationship, which is the time required to complete the work. But for computers, there are several parameters that affect the completion speed, such as the speed of the central processor, memory requirements, RAM, bandwidth, path length to, and from the devices, as well as possible path congestion [5], [9]. In this paper, we used some of these parameters with the Johnson procedure and apply it to several servers. As a result, it is possible to get the desired sequence by which the overlap of servers is optimal and suitable for application in a computing environment.

2. METHOD

2.1. Principal assumptions

The general assumptions for sequence problems are: i) know the processing times on each server, ii) the time required to complete the task is independent of the order of the functions to be processed, iii) no servers can process more than one function at the same time, and iv) transition time from one server to another is negligible. Every task is non-preemptive (once it starts the execution on the server, we cannot interrupt it until the execution finished). When several functions given to be executed and require processing on two or more servers, the major concern of the manager is to find the order or sequence to perform these functions [3], [10]–[12]. We can broadly divide these sequence problems into two groups.

In the first case, there are N number of tasks to be performed, each of which requires processing on some or all different k servers. We can determine the effectiveness of both sequences that are technically workable (i.e. those that meet the limitations of the order in which each function must be processed through machines) and choose a sequence that improves effectiveness. For clarity, we may specify the processing times of each of the N functions on each of the k machines. In a certain order and the performance time of the functions may be the measure of effectiveness. We will choose sequences in which the total time spent processing all functions on the devices is minimal. The possibilities that could include the number of tasks and the number of servers is: i) two machines A and B have N functions, and each function will go to machine A first and then to B , ii) three machines A , B , and C have N functions, and each function will go to machine A first and B second and then to C , and iii) problems with n jobs and m machines. All of the above possibilities can take the same approach used in this research paper, where we have explained the first case in (3.2) and in the same way that the other second and third cases can have adopted.

2.1.1. Handling N number of tasks on two devices

The easier possible sequence problem is the N -machine sequence problem. Where we want to determine which sequence n -job should be handled by two machines to reduce the total elapsed time T [13]. The matter can be qualified as: i) two devices were used A and B only, ii) each function is implemented in a sequence of AB , and iii) right or predictable times of implementation of all jobs recognized and given in the following Table 1.

Table 1. Expected processing times

Table 1: Expected processing times									
Machine	Job(s)								
	1	2	3	--	-	i	--	-	n
A	A ₁	A ₂	A ₃	--	-	A _i	--	-	A _n
B	B ₁	B ₂	B ₃	--	-	B _i	--	-	B _n

2.2. Johnson procedure steps

The dilemma is to find the order of functions to reduce the amount of elapsed time T . We know the solution to this dilemma is the Johnson procedure [1] which includes the following steps:

- Step 1: select the lowest processing time occurring in the list of jobs, if there is a tie, select any one.
- Step 2: if the minimum processing time is A_r , select the r th function first. If the minimum processing time is B_s , select the s th function last. because the given order is AB .

- c. Step 3: repeat the first and second steps for the remaining set of processing times obtained by omitting the processing time for each of the two machines corresponding to the already assigned task.
- d. Step 4: continue in the same way until all functions are processed.
- e. Step 5: find the idle times and the sum of elapsed time on devices A and B, then the result is an optimal sequence.

Total elapsed time= $T_1 - T_2$

Where T_1 = the starting time of the first task in the optimum order on the first device, and T_2 = the completing time of the last process in the optimum order on the second device.

Idle time on machine A = $T_2 - (T_1 + \text{turnaround time of the first task})$.

Machines performance: there are five most common causes that cause a bottleneck.

- CPU utilization

CPU usage can increase significantly. In particular, if each entity can interact with each other in the same space, the load will increase quadratically with the number of entities [14]. When the system is overloaded, the rate to send updates to the users is slowing down, and responses to user input, can lead to a significant deterioration in user experiences.

Figure 1 shows examples of CPU usage with different loads on different actors. The increase in load is directly proportional to the increase in the number of simultaneous users. As the number of simultaneous users increases, the CPU's quadratic load will increase around $O(n^2)$. With the rise in the grist of simultaneous users, the network can flood the traffic with a square speed that coincides with the number of concurrent users. According to Microsoft, "if the processor is too busy and cannot respond to all requests within that time, then there will be bottlenecks in the processor". It therefore cannot to perform its assigned tasks in a timely manner. This bottleneck appears in two cases: the first is when the processor is working at over 80% of its capacity for a long time, and the second is when the queue is extremely long.

- Memory utilization

As shown in Figure 1, the computer memory bottlenecks indicate that the system does not have fast RAM and/or does not have enough one [15]. Then the speed of data delivery to the CPU will reduce, as a result, the slow implementation of the processes. Where the system does not have enough will transfer the stored data to the hard drive through the swap-out and swap-in process to keep the processes running, which significantly slows down the system. If the RAM cannot connect data to the CPU quickly enough, the device will experience a slowdown, and therefore the CPU will enter the idle phase most of the time [16].

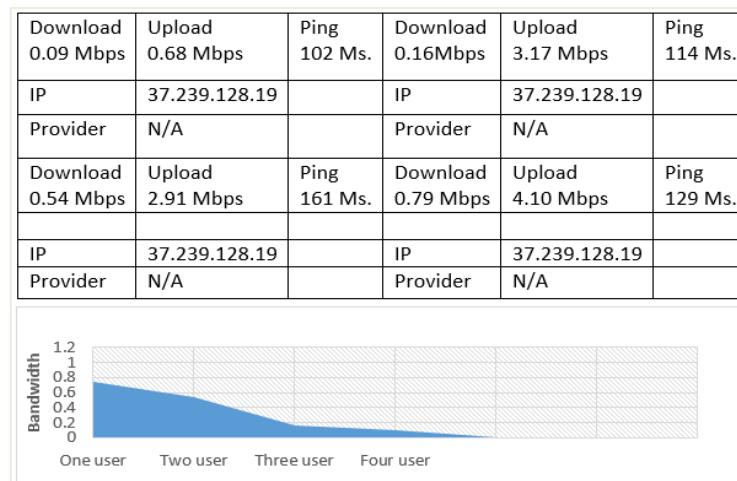


Figure 1. Bandwidth and CPU utilization vs number of simultaneous users

- Network utilization

If the required bandwidth is insufficient for inter-device communication or processing power to complete the task quickly enough, network bottlenecks will occur [17], [18]. It is also unnecessary to use bandwidth more than the actual need. According to Microsoft, bottlenecks on the network occur when the network loses its integrity and when there is an overload on the server, or on the network connection device. If we ask, how much bandwidth do we really need, anyway? ISPs, have the simplest answer to this question, there must be a balance between price and performance and the result is the best option. Determining the

level of performance that you need is directly related to the type of service required, as prices vary from place to place for service providers, and this requires us to make some concessions in one option at the expense of another. So, first, we need to know the speed we need to support the devices used on the job site [19]. A service offering that provides a range of megabits per second of download bandwidth should provide a good level of performance that supports all the current optimal use cases for most people working on these devices.

- Software limitation

Sometimes, identifying bottlenecks due to the program itself. Programs handle a few tasks simultaneously, so that the program does not use any additional CPU or RAM assets, even when they are available. In addition, we write the program in a manner not compliant with multiple CPU streams [20]. Therefore; it used only one core even if the processor is multi-core.

- Disk usage

Typically, long-term storage is the slowest component inside a computer. This includes a hard disk drives and SSDs, often a bottleneck in computers. Even with faster long-term storage, its actual speed is limited [21].

2.2.1. Example of Johnson method

Suppose we have seven jobs. All of them must pass to servers A and B in the order AB, and the processing time in milliseconds shown as in Table 2. To determine the order of these functions that will minimize the total elapsed time T. We find the smallest job is 1 ms for job 6 on machine B so we have to schedule job 6 last on machine A as shown in Table 3. The processing time reduced set becomes as shown in Table 4.

Table 2. Processing time

Job number \ Server	1	2	3	4	5	6	7
Server A	3	12	15	6	10	11	9
Server B	8	10	10	6	12	1	3

Table 3. Schedule task number 6

Job number	
Job sorting	6

Table 4. Updating processing time

Job number \ Server	1	2	3	4	5	---	7
Server A	3	12	15	6	10	---	9
Server B	8	10	10	6	12	---	3

There are two equal minimum values processing time of 3 ms for job 1 on machine A and for job 7 on machine B, according to the rules, job 1 is scheduled first and job 7 next to job 6 as shown in Table 5. After that, Repeat the procedure until the end, so the sequence will be as shown in Table 6. Thus, the final result that can be obtained through the work of both servers can be obtained in Table 7. The minimum elapsed time is 67 ms, the idle time for server A is 1 (67-66), and the idle time for server B is 17 ms. If the processing time is changed by another representing the bandwidth or memory capacity, the string will change to a different one.

Table 5. Schedule task number 1, 6, 7

Job number		
Job sorting	1	7 6

Table 6. Schedule all the tasks

Job number						
Job sorting	1	4	5	3	2	7 6

Table 7. Idle time for servers A and B

Jobs \ Servers	Server A		Server B		Idle time for server B
	Time in	Time out	Time in	Time out	
1	0	3	3	11	3
4	3	9	11	17	0
5	9	19	19	31	2
3	19	34	34	44	3
2	34	46	46	56	2
7	46	55	56	59	0

6	55	66	66	67	7
---	----	----	----	----	---

2.3. Proposed strategy

While all parameters are important and cannot neglect at the expense of a single parameter with load balancing [22]–[25], the distribution of loads relied on calculating memory. For example, as in the weighted Round Robin algorithm, the weights representing memory capacity are the target in the number of requests sent to a particular server. In this paper, we have proposed a simple method to integrate most of all these requirements in a certain percentage that is determined by the admin, as shown in (1):

$$\text{Let result } i = P_{iBW} \times W_{iBW} + P_{iCPU} \times W_{iCPU} + P_{iI/O} \times W_{iI/O} \text{ and } W_{iBW} + W_{iCPU} + W_{iI/O} = 1 \quad (1)$$

Where result i represents the cost of selecting the required model for the service i where $1 \leq i \leq n$, P_{iBW} represents the percentage of the available bandwidth between the client and the server i , which should be divided by the higher theoretical bandwidth, W_{iBW} represents the percentage of network and server bandwidth i determined by the administrator of the organization, P_{iCPU} represents the CPU idle state of server i , W_{iCPU} represents the percentage of CPU of server i determined by the administrator of the data grid organization, $P_{iI/O}$ represents the memory-free space of server i , and $W_{iI/O}$ represents the percentage of memory-free space determined by the administrator of the data grid organization.

2.4. Proposed algorithm

The following algorithm is implemented using the C++ language; this algorithm is used to implement two functions (A and B), the output of one of which is the input of the other. This algorithm is supported by some parameters; the first is the CPU speed and memory capacity, which slightly affects the speed of task execution; the second parameter is the value of the bandwidth between servers and clients; this value is the most important factor affecting the speed of task completion. The purpose of implementing the algorithm is to reduce the idle time of the processor and thereby reduce the execution time.

```

Start
WiBW + WiCPU + WiI/O = 1;
for all server; Read CPU speed and Memory capacity;
for all servers; Read the bandwidth amount between servers and clients;
for all servers; Read the weight of CPU speed, Memory capacity, and bandwidth;
For all i of first server A;
A[i]=result  $i = P_{iBW} \times W_{iBW} + P_{iCPU} \times W_{iCPU} + P_{iI/O} \times W_{iI/O}$ ; // i=number of job
For all i of second server B;
B[i]=result  $i = P_{iBW} \times W_{iBW} + P_{iCPU} \times W_{iCPU} + P_{iI/O} \times W_{iI/O}$ ;
For all i; Read minimum value of A[i] and minimum value of B[i];
If A[i] < B[i];
K[i]=A[i] else K[i]=B[i]; // K[i]=scheduling array
Remove A[i] and B[i] from the list;
Continue until the list become empty;
End

```

3. RESULTS AND DISCUSSION

Suppose that a certain agent works to provide services to its customers. Through his experience, the requirements for CPU speed are 70% important while the memory capacity is 30%, but the bandwidth size is 10% of what is available is enough. Table 8 illustrates the process time, bandwidth, and memory capacity between each job (clients) and server A.

Table 8. The process time, bandwidth, and memory capacity between each client and server A

Server specification \ Job	1	2	3	4	5	6	7
CPU speed	3	12	15	6	10	11	9
bandwidth	5	4	7	8	5	3	10
Memory capacity in GB	200	400	800	500	100	600	300
Result $i = P_{iBW} \times W_{iBW} + P_{iCPU} \times W_{iCPU} + P_{iI/O} \times W_{iI/O}$	62.6	128.8	251.2	155	37.5	188	96.4

Table 9 illustrates the CPU speed in gigahertz, bandwidth, and memory capacity in GB between each job (clients) and server B. Now we have values that differ from those in the example, which depend on the processing speed. When we use these values and do the same processing above, the chain that is best to follow is as follows and the result of implementation of the Jonson method is as shown in Table 10, while the work of both servers together can be illustrated by Table 11.

Table 9. Processing time

Job \ Server	1	2	3	4	5	6	7
Server A	62.5	128.8	251.2	155	37.5	188	96.4
Server B	35.7	91.5	211.4	125	38.8	180.9	123

Table 10. Tasks schedule

	Execution sequence						
Tasks	5	7	3	6	4	2	1

Table 11. Idle time for servers A and B

Jobs \ Server	Server A		Server B		Idle time for server B
	Time in	Time out	Time in	Time out	
5	0	10	10	5	0
7	10	19	22	7	10
3	19	34	34	3	19
6	34	45	45	6	34
4	45	51	51	4	45
2	51	63	63	2	51
1	63	66	73	1	63

The minimum elapsed time is 81 ms (it is the time when all tasks are completed by both servers), the idle time for server A is 15 (the difference between the server's end time $=81-66=15$), and the idle time for server B is 31 ms. Figure 2 illustrates the processing time for all the jobs by each server individually, while Figure 3 shows the overlap times between servers. Figures 4 and 5 shows the processing time, bandwidth, and memory capacity between each client and servers A and B respectively. Servers A's and B's processing time for each task is shown in Table 9 and Figure 6 shows the relationship between these two servers based on the score value of each.

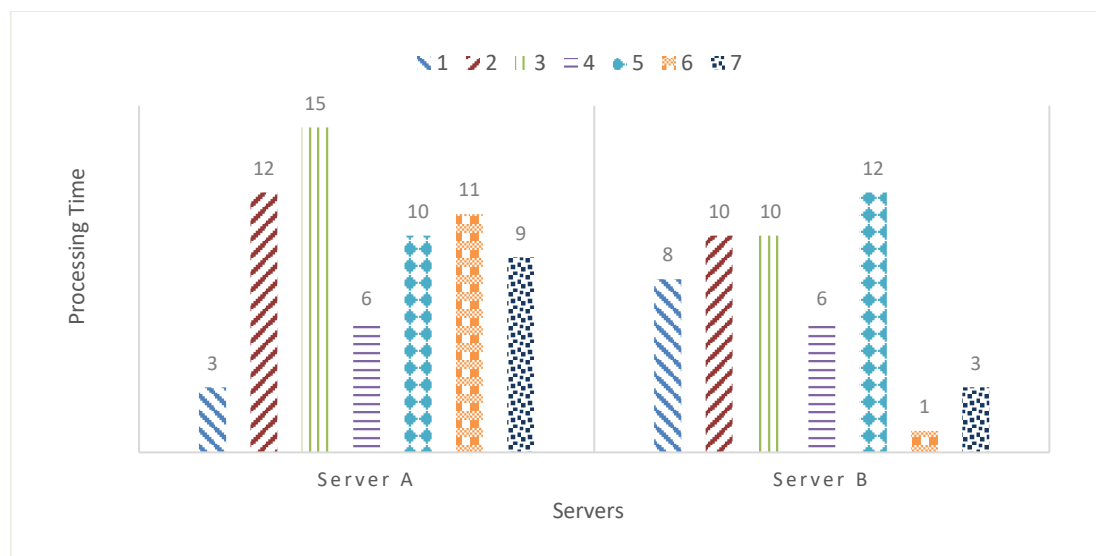


Figure 2. Process time for all jobs

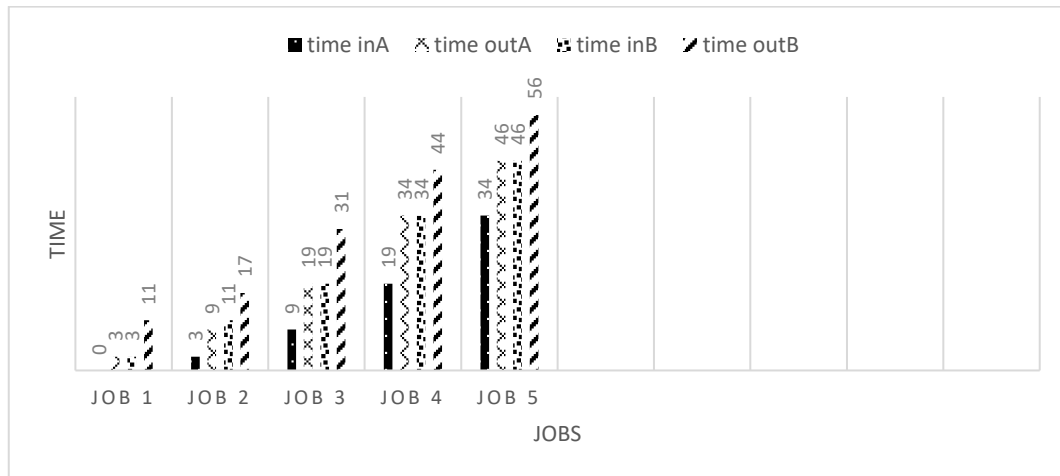


Figure 3. Interference times between servers

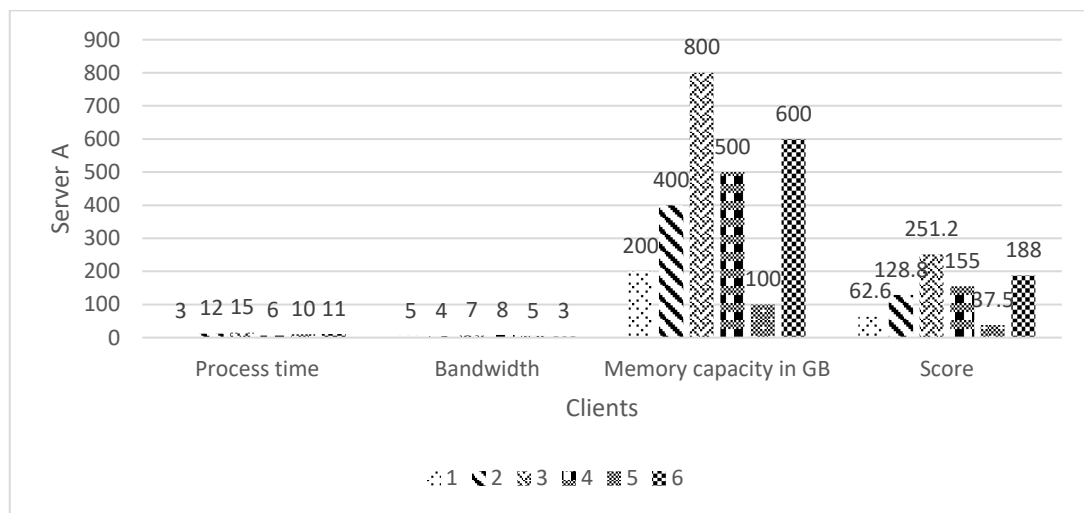


Figure 4. The processing time, bandwidth, and memory capacity between each job (clients) and server A

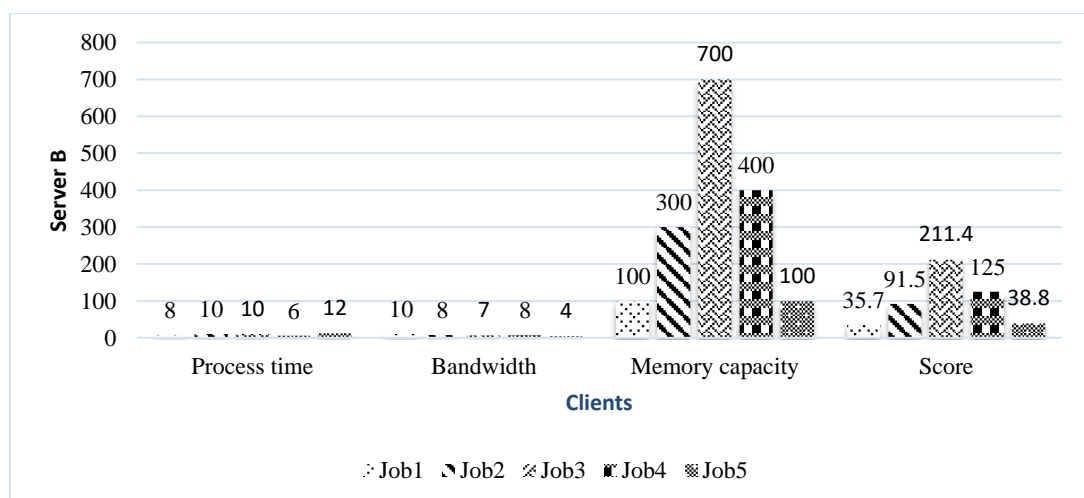


Figure 5. Score of the process time, bandwidth, and memory capacity between each clients and server B

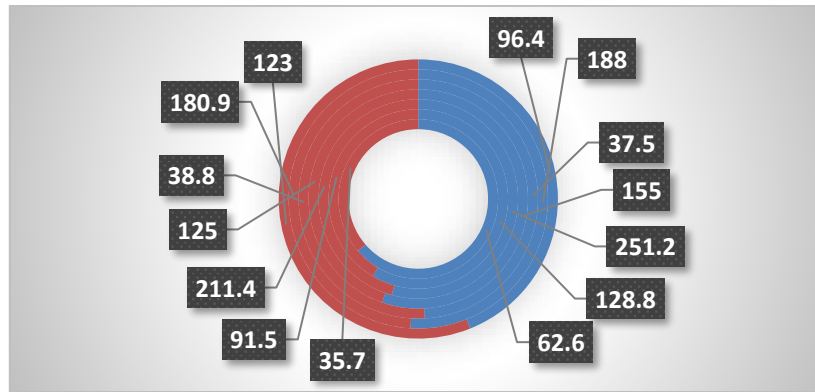


Figure 6. The process time, bandwidth, and memory capacity between each client and servers

4. CONCLUSION

Through the example, we found that the idle time of both servers is higher with entering other values such as bandwidth and memory that may be desired in the calculations such as the size of the RAM or the speed of the network. But in fact the application from the practical side is contrary to this because the theoretical calculations do not reflect the reality of the situation. For example, calculating the values for the processing speed of the server is inside the machine only, while the data may not reach it due to poor internet, lack of bandwidth, or low speed in the network itself, and perhaps due to high traffic. These reasons may lead to a very high idle time if the rest of the parameters are not considered.

ACKNOWLEDGEMENTS

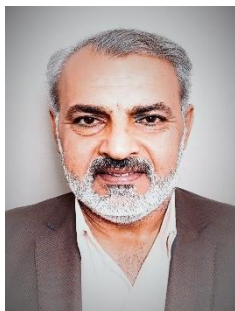
We extend our thanks and appreciation to the University of Babylon and the College of Information Technology for their continuous support of the staff.




REFERENCES

- [1] S. M. Johnson, "Optimal two-and three-stage production schedules with setup times included," *Naval Research Logistics Quarterly*, vol. 1, no. 1, pp. 61–68, 1954, doi: 10.1002/nav.3800010110.
- [2] M. Koehler and S. Benkner, "A service oriented approach for distributed data mediation on the Grid," *8th International Conference on Grid and Cooperative Computing, GCC 2009*, pp. 401–408, 2009, doi: 10.1109/GCC.2009.35.
- [3] S. A. Abbas and M. S. Almhanna, "Distributed denial of service attacks detection system by machine learning based on dimensionality reduction," *Journal of Physics: Conference Series*, vol. 1804, no. 1, pp. 1–12, 2021, doi: 10.1088/1742-6596/1804/1/012136.
- [4] N. E. Y. Kouba, M. Menaa, M. Hasni, and M. Boudour, "Load frequency control in multi-area power system based on fuzzy logic-PID controller," in *2015 IEEE International Conference on Smart Energy Grid Engineering (SEGE)*, 2015, pp. 1–6, doi: 10.1109/SEGE.2015.7324614.
- [5] M. S. Almhanna, "Minimizing replica idle time," in *2017 Annual Conference on New Trends in Information and Communications Technology Applications, NTICT 2017*, 2017, pp. 128–131, doi: 10.1109/NTICT.2017.7976134.
- [6] B. Cornelis, "Site autonomy in a distributed database environment," in *Digest of Papers. COMPCON Spring 88 Thirty-Third IEEE Computer Society International Conference*, 1988, pp. 440–443, doi: 10.1109/COMPCON.1988.4908.
- [7] R. L. Chakrasali, H. N. Nagaraja, B. S. Shavaladi, and V. R. Sheelavant, "Rural load management using information technology," *Bulletin of Electrical Engineering and Informatics*, vol. 1, no. 3, pp. 199–202, 2012, doi: 10.12928/eei.v1i3.236.
- [8] H. K. Omar, K. H. Jihad, and S. F. Hussein, "Comparative analysis of the essential cpu scheduling algorithms," *Bulletin of Electrical Engineering and Informatics*, vol. 10, no. 5, pp. 2742–2750, 2021, doi: 10.11591/eei.v10i5.2812.
- [9] N. Todorov, I. Ganchev, and M. O'Droma, "Exploring the congestion level index for defining the QoS performance profile of internet paths," in *2020 28th National Conference with International Participation (TELECOM)*, 2020, pp. 97–100, doi: 10.1109/TELECOM50385.2020.9299545.
- [10] R. M. Almuttairi, R. Wankar, A. Negi, R. R. Chillarige, and M. S. Almahna, "New replica selection technique for binding replica sites in data grids," in *EPC-IQ01 2010 - 2010 1st International Conference on Energy, Power and Control*, 2010, pp. 187–194, doi: 10.37917/ijeee.6.2.16.
- [11] R. M. Almuttairi, R. Wankar, A. Negi, and C. R. Rao, "Enhanced data replication broker," in *Multi-disciplinary Trends in Artificial Intelligence: 5th International Workshop*, 2011, pp. 286–297, doi: 10.1007/978-3-642-25725-4_25.
- [12] K. H. Anun and M. S. Almhanna, "Web server load balancing based on number of client connections on docker swarm," in *Proceedings of 2021 2nd Information Technology to Enhance E-Learning and other Application Conference, IT-ELA 2021*, 2021, pp. 70–75, doi: 10.1109/IT-ELA52201.2021.9773748.
- [13] A. Attanasio, G. Ghiani, L. Grandinetti, and F. Guerriero, "Auction algorithms for decentralized parallel machine scheduling," *Parallel Computing*, vol. 32, no. 9, pp. 701–709, 2006, doi: 10.1016/j.parco.2006.03.002.
- [14] H. Liu, M. Bowman, R. Adams, J. Hurliman, and D. Lake, "Scaling virtual worlds: simulation requirements and challenges," in *Proceedings - Winter Simulation Conference*, 2010, pp. 778–790, doi: 10.1109/WSC.2010.5679112.




- [15] L. D. Paulson, "Faster RAM tackles data and marketplace bottlenecks," *Computer*, vol. 35, no. 3, pp. 17–19, 2002, doi: 10.1109/2.993766.
- [16] M. Hans and V. Jogi, "Peak load scheduling in smart grid using cloud computing," *Bulletin of Electrical Engineering and Informatics*, vol. 8, no. 4, pp. 1525–1530, 2019, doi: 10.11591/eei.v8i4.1919.
- [17] A. A. Taiwo, K. R. K.-Mahamud, and M. S. b. Sajat, "Locating bottleneck nodes on a large wired local area network," in *2011 National Postgraduate Conference*, 2011, pp. 1–5, doi: 10.1109/NatPC.2011.6136343.
- [18] S. Dublisch, V. Nagarajan, and N. Topham, "Evaluating and mitigating bandwidth bottlenecks across the memory hierarchy in GPUs," in *2017 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2017, pp. 239–248, doi: 10.1109/ISPASS.2017.7975295.
- [19] A. Saoud and A. Recioui, "Hybrid algorithm for cloud-fog system based load balancing in smart grids," *Bulletin of Electrical Engineering and Informatics*, vol. 11, no. 1, pp. 477–487, 2022, doi: 10.11591/eei.v11i1.3450.
- [20] S. S. Jalali, H. Rashidi, and E. Nazemi, "A new approach to evaluate performance of component-based software architecture," in *2011 UKSim 5th European Symposium on Computer Modeling and Simulation*, 2011, pp. 451–456, doi: 10.1109/EMS.2011.77.
- [21] B. Liskov and L. Shrira, "Escaping the disk bottleneck in fast transaction processing," in *3rd Workshop on Workstation Operating Systems, WWOS 1992*, 1992, pp. 118–121, doi: 10.1109/WWOS.1992.275677.
- [22] Y. Zhang and N. Lu, "Parameter selection for a centralized thermostatically controlled appliances load controller used for intra-hour load balancing," *IEEE Transactions on Smart Grid*, vol. 4, no. 4, pp. 2100–2108, 2013, doi: 10.1109/TSG.2013.2258950.
- [23] H. J. Abd and M. S. Almahanna, "Suppression of a nonlinear effect for high data transmission rate systems with a wavelength division multiplexer using the optimization of fiber properties," *Ukrainian Journal of Physics*, vol. 62, no. 7, pp. 583–588, 2017, doi: 10.15407/ujpe62.07.0583.
- [24] R. Kaur and G. Kaur, "Proactive scheduling in cloud computing," *Bulletin of Electrical Engineering and Informatics*, vol. 6, no. 2, pp. 174–180, 2017, doi: 10.11591/eei.v6i2.649.
- [25] M. V. Gopalachari, P. Sammulal, and A. V. Babu, "Correlating scheduling and Load balancing to achieve optimal performance from a cluster," in *2009 IEEE International Advance Computing Conference*, 2009, pp. 320–325, doi: 10.1109/IADCC.2009.4809029.

BIOGRAPHIES OF AUTHORS






Mahdi S. Almhanna    received the Ph.D. in Computer Science from Osmania University, India. Over the years, he was assigned the duties of the head of the Department of Computer Technology at the College of Almustaqbal and supervised several M.Sc. students. Currently, he is associate professor in Department of Information Security at University of Babylon, Babylon, Iraq. He can be contacted at email: mahdi.almhanna@uobabylon.edu.iq or mahdialmhanna@gmail.com.



Firas Sabah Al-Turaihi    is a lecturer at University of Babylon, College of Information Technology, and Networks Departments. He received his PhD degree from Brunel University London, United Kingdom. He received his BSc and MSc degrees in Computer Science. His research interests cover communication networks. He can be contacted at email: firassabahalturaihi@uobabylon.edu.iq.



Tariq A. Murshedi    received the Ph.D from Northeastern University, Shenyang, China. He is also an instructor in Cisco Networking Academy, Babylon University, Iraq. His research interests include routing protocols in mobile ad hoc networks and wireless network. Currently, he is lecture in Department of Information Networks at University of Babylon, Babylon, Iraq. He can be contacted at email: tariq_alwan@itnet.uobabylon.edu.iq.